
Space-time discretization of the heat equation

A concise Matlab implementation

Roman Andreev

December 27, 2012

Abstract A concise Matlab implementation of a stable parallelizable space-time Petrov-Galerkin discretization for parabolic evolution equations is given. Emphasis is on the reusability of spatial finite element codes.

Keywords heat equation · parabolic · space-time discretization · parallel · preconditioning · Matlab · implementation

1 Introduction

The spectrum of numerical methods for parabolic evolution equations is extremely broad, which attests to the ubiquity and the relevance of such equations. With the aim of developing reliable massively parallel algorithms, e.g. for optimization problems constrained by parabolic evolution equations, several attempts have been made to go beyond time-stepping methods, see [3, Section 5.1] for a modest attempt of an overview. In this paper we give a concise Matlab implementation of a specific space-time Petrov-Galerkin discretization for parabolic evolution equations from [4, 3], hoping to provide a basis for possible further developments. Spanning just a few lines of Matlab code, it is parallelizable and stable in the Petrov-Galerkin sense, which already distinguishes it from conventional methods for parabolic evolution equations. Stability implies quasi-optimality of the discrete solution in the natural solution spaces, and is an important property in the resolution of nonlinear problems. Moreover, the implementation is modular with respect to the spatial discretization, admits time-dependent inputs and nonuniform temporal grids. Since the algorithm is based on an iterative solution of a *single* linear system, another significant advantage to conventional methods is the ability to terminate the iteration at a specified global accuracy.

The model parabolic evolution equation under consideration is presented in Section 2 and is restated in a variational formulation. The space-time Petrov-Galerkin discrete trial and test spaces that will be used to discretize the variational

R. Andreev
Seminar for Applied Mathematics, Rämistr. 101, 8092 Zürich, Switzerland
E-mail: andreevr@math.ethz.ch

formulation are introduced in Section 3. In order to obtain stable algorithms we develop in Section 4 a generalization of the usual variational framework for linear operator equations, called the minimal residual Petrov-Galerkin discretization. Choosing bases on the discrete trial and test spaces, it leads to a linear system of generalized Gauß normal equations along with a natural preconditioner. In that framework, certain norm-inducing operators play an important role. Specifically for the parabolic evolution problem, these are defined in Section 5. In Section 6 we detail the Kronecker product structure of the parabolic space-time operator and the norm-inducing operators when assembled using space-time tensor product bases, and comment on the data structures employed. In the solution process, the inverses of the matrix representations (of norm-inducing operators) are required. Their Kronecker product structure is discussed in Section 7. The assembly procedure for the space-time source is in Section 8. A generalization of the LSQR algorithm of Paige and Saunders [16] for the iterative resolution of the linear system is given in Section 9. With those preparations, the Matlab implementation is presented in Section 10, and a numerical example is given. We conclude and point out some limitations in Section 11.

2 Model problem and its space-time variational formulation

Let $D \subset \mathbb{R}^n$, $n \in \{1, 2, 3\}$, be a bounded connected domain with a polyhedral boundary $\Gamma = \partial D$. If $n = 1$ then D is an open bounded interval; if $n = 2$ then D is a polygon; if $n = 3$ then D is a polyhedron. We partition Γ into two disjoint subsets Γ_0 and Γ_N such that $\bar{\Gamma} = \bar{\Gamma}_0 \cup \bar{\Gamma}_N$. On Γ_0 we will impose homogeneous boundary conditions of Dirichlet type. For that reason, the Dirichlet boundary Γ_0 is assumed to be of positive measure (with respect to the surface measure which will subsequently be denoted by σ), i.e., it contains at least one endpoint if $n = 1$; a curve of positive length if $n = 2$; or a surface of positive surface measure if $n = 3$. Let $J = (0, T)$, $T > 0$, denote the temporal interval.

The model for parabolic evolution equations that we consider is the heat equation:

$$\partial_t u(t, x) - \operatorname{div}(a(x) \operatorname{grad} u(t, x)) = f(t, x), \quad (t, x) \in J \times D, \quad (1)$$

$$u(t, x) = 0, \quad (t, x) \in J \times \Gamma_0, \quad (2)$$

$$a(x) \frac{\partial u}{\partial n}(t, x) = g(t, x), \quad (t, x) \in J \times \Gamma_N, \quad (3)$$

$$u(t, x) = h(x), \quad (t, x) \in \{0\} \times D. \quad (4)$$

Here, a , f , g and h are given scalar valued functions, while u is the unknown; further, $\Delta = \sum_{i=1}^n \frac{\partial^2}{\partial x_i^2}$ and $\frac{\partial}{\partial n}$, respectively, denote the Laplace operator on D and the derivative in the direction of the outward normal at the Neumann part Γ_N of the boundary. The precise meaning of the heat equation will be fixed by means of a well-posed space-time variational formulation in the following. The time-independent operator “ $\operatorname{div}(a(x) \operatorname{grad} u(t, x))$ ” could be replaced by the time-dependent one “ $\operatorname{div}(\mathbf{A}(t, x) \operatorname{grad} u(t, x))$ ”, where $\mathbf{A} \in L^\infty(J \times D; \mathbb{R}_{\operatorname{sym}}^{n \times n})$, without affecting most considerations below (the technical reason why this is possible is given in [13, Lemma 4.4.1]). However, if \mathbf{A} is not a finite sum of separable functions, the implementation becomes significantly less transparent, and we therefore

discard this case from the onset on; on the other hand, \mathbf{A} being a finite sum of separable functions entails modifications of secondary relevance to this exposition.

To motivate the space-time variational formulation of the heat equation, we formally test the equation with a function v_1 on $J \times D$ and integrate in space and time; the initial condition is tested with v_2 on D and integrated in space. Integration by parts is performed in space only, and the two resulting conditions (one “ $\forall v_1$ ”, the other “ $\forall v_2$ ”) are added together. The solution u will be sought in the space X , and the test functions are combined to $v = (v_1, v_2) \in Y := Y_1 \times Y_2$. The spaces X and Y will be specified presently. We write $(\cdot, \cdot)_D$ for the $L^2(D)$ and the $[L^2(D)]^d$ scalar product, while $(\cdot, \cdot)_{\Gamma_N}$ is the scalar product on $L^2(\Gamma_N)$ for the boundary measure σ introduced above. Generally, we omit the dependence of the integrands on the temporal variable t . In this way we obtain the continuous space-time variational formulation

$$\text{find } u \in X \text{ such that } B(u, v) = b(v) \quad \forall v \in Y, \quad (5)$$

where the system bilinear form B encoding the heat equation is

$$B(u, v) := \int_J (\partial_t u, v_1)_D dt + \int_J (a \operatorname{grad} u, \operatorname{grad} v_1)_D dt + (u(0, \cdot), v_2)_D \quad (6)$$

and the load functional b supplying the source term, as well as boundary and initial data, is

$$b(v) := \int_J (f, v_1)_D dt + \int_J (g, v_1)_{\Gamma_N} dt + (h, v_2)_D. \quad (7)$$

Integrating by parts also in time, as in e.g. [6, 7], leads to an alternative space-time variational formulation, which, however, will not be discussed below.

Let us introduce the abbreviations

$$V := H_{\Gamma_0}^1(D), \quad H := L^2(D) \quad \text{and} \quad V' = [H_{\Gamma_0}^1(D)]', \quad (8)$$

where $H_{\Gamma_0}^1(D)$ denotes the Sobolev space of functions in $H^1(D)$ with vanishing trace on the Dirichlet boundary Γ_0 , and $[H_{\Gamma_0}^1(D)]'$ its dual. We identify H with its dual H' by means of the Riesz isomorphism on H . Then the duality pairing on $V \times V'$ (or $V' \times V$) is the continuous extension of the H -scalar product on $V \times V$. In this way we obtain a so-called Gelfand triple of separable Hilbert spaces

$$V \hookrightarrow H \cong H' \hookleftarrow V' \quad (9)$$

with continuous and dense embeddings. In order for $B : X \times Y \rightarrow \mathbb{R}$ to be a continuous bilinear form and for $b : Y \rightarrow \mathbb{R}$ to be a continuous linear functional, it is now natural to take

$$X := L^2(J; V) \cap H^1(J; V') \quad \text{and} \quad Y := L^2(J; V) \times H. \quad (10)$$

For the definition of the Bochner spaces $L^2(J; V)$, $H^1(J; V')$, and the like, we refer to e.g. [12] or [15, Chapter 1]. Then (6)–(7) are well-defined for all $(u, v) \in X \times Y$ whenever

$$f \in L^2(J; [H_{\Gamma_0}^1(D)]'), \quad g \in L^2(J; H^{1/2}(\Gamma_N)) \quad \text{and} \quad h \in L^2(D), \quad (11)$$

and

$$a \in L^\infty(D) \quad \text{with} \quad 0 < \text{ess inf } a \leq \text{ess sup } a < \infty. \quad (12)$$

For the remainder of the article we assume $f \in L^2(J; L^2(D))$. The spaces X and Y are themselves Banach spaces for the norms $\|\cdot\|_X$ and $\|\cdot\|_Y$ that are given by

$$\|u\|_X^2 := \|u\|_{L^2(J; V)}^2 + \|\partial_t u\|_{L^2(J; V')}^2, \quad u \in X, \quad (13)$$

$$\|v\|_Y^2 := \|v_1\|_{L^2(J; V)}^2 + \|v_2\|_H^2, \quad v = (v_1, v_2) \in Y. \quad (14)$$

We recall from e.g. [12, Section 5.9.2] or [15, Chapter 1] that any (representant of any) $u \in X$ admits a modification on a negligible subset of J such that the resulting function coincides with a unique continuous H -valued function defined on the closed interval \bar{J} ; moreover, the C^0 norm of the latter is controlled by the X norm of the former. In other words, the following embedding is continuous

$$X = L^2(J; V) \cap H^1(J; V') \hookrightarrow C^0(\bar{J}; H), \quad (15)$$

and in particular

$$\exists C > 0 : \quad \|u(0)\|_H \leq C \|u\|_X \quad \forall u \in X. \quad (16)$$

In this way, the initial value $u(0)$ of any $u \in X$ is well-defined in H .

With the above assumptions, the space-time variational problem (5) has a unique solution $u \in X$ and the solution depends continuously on the functional $b \in Y'$, see [17, Theorem 5.1] and the references therein. Hence, the solution u also depends continuously on the input data f , g and h that define the load functional b in (7).

3 Space-time tensor product discrete trial and test spaces

The continuous space-time variational formulation (5) will be discretized using finite-dimensional discrete trial and test spaces $X_h \subset X$ and $Y_h \subset Y$ built up from finite-dimensional “univariate” temporal subspaces $E \subset H^1(J)$, $F \subset L^2(J)$, and spatial subspaces $V_h \subset V$. These spaces assume the space-time tensor product form

$$X_h := E \otimes V_h \quad \text{and} \quad Y_h := (F \otimes V_h) \times V_h. \quad (17)$$

A key feature of this discretization and the implementation given below is the modularity with respect to the spatial subspaces V_h .

To specify the temporal subspaces E and F we need to introduce some terminology. A temporal mesh \mathcal{T} is a finite set of points in $\bar{J} = [0, T]$ containing 0 and T . The connected components of $J \setminus \mathcal{T}$ are called the elements of \mathcal{T} . Let $\max \Delta \mathcal{T}$ denote the maximal “time-step”, i.e., the maximal length of an element of \mathcal{T} . For a temporal mesh \mathcal{T} let \mathcal{T}^* denote the temporal mesh obtained from \mathcal{T} by a uniform refinement (each element is split into two smaller elements of equal length). Concerning E on the trial side and F on the test side, we will restrict ourselves to two types of pairs that differ in the choice of F .

Type 1 temporal subspaces. Given a temporal mesh \mathcal{T}_E , we define E as the standard space of continuous piecewise affine functions, and F as the space of piecewise constant functions on $\mathcal{T}_F := \mathcal{T}_E$.

Type 2 temporal subspaces. Given a temporal mesh \mathcal{T}_E , the space E is defined as above. Let another temporal mesh, \mathcal{T}_F be obtained from \mathcal{T}_E by a succession of uniform refinements, i.e., $\mathcal{T}_F = [\mathcal{T} \mapsto \mathcal{T}^*]^n(\mathcal{T}_E)$ for some positive $n \in \mathbb{N}$. Then F is defined as the space of piecewise constant functions on \mathcal{T}_F .

In the first case the discrete variational formulation

$$\text{find } u_h \in X_h \text{ such that } B(u_h, v_h) = b(v_h) \quad \forall v_h \in Y_h \quad (18)$$

is an example of continuous Galerkin time-stepping schemes [14, 1]. In the second case the dimension of Y_h is larger than that of X_h , and the above discrete variational formulation is meaningless. A generalization based on residual minimization is therefore introduced in Section 4. Concerning the stability of the resulting minimal residual Petrov-Galerkin method there is a fundamental difference between Type 1 and Type 2 temporal subspaces. This is the subject of the following two propositions that summarize the relevant main results from [3, Section 5.2.3]. Note carefully that the present concept of stability, namely the validity of the discrete inf-sup condition

$$\gamma_h := \inf_{u_h \in X_h \setminus \{0\}} \sup_{v_h \in Y_h \setminus \{0\}} \frac{B(u_h, v_h)}{\|u_h\|_X \|v_h\|_Y} > 0 \quad (19)$$

(its role is discussed in Section 4) uniformly in the choice of the temporal discretization, is different from e.g. A-stability for time-stepping methods.

The following measure of self-duality for the spatial subspace $V_h \subset V$ will be needed:

$$\kappa_h := \inf_{\chi'_h \in V_h \setminus \{0\}} \sup_{\chi_h \in V_h \setminus \{0\}} \frac{(\chi'_h, \chi_h)_D}{\|\chi'_h\|_{V'} \|\chi_h\|_V}. \quad (20)$$

Note that κ_h is bounded (independently of V_h) and necessarily positive for a finite-dimensional V_h .

Proposition 1 *Let $\{0\} \neq V_h \subset V$ be a finite-dimensional subspace. Let $E \subset H^1(J)$ and $F \subset L^2(J)$ be of Type 2. Then there exists a constant $\gamma_0^* > 0$ independent of V_h , E and F , such that the discrete inf-sup condition (19) holds for the discrete trial and test spaces (17) with*

$$\gamma_h \geq \gamma_0^* \kappa_h. \quad (21)$$

We remark that γ_0^* in (21), as a function of the number of refinements between \mathcal{T}_E and \mathcal{T}_F , is monotonically increasing and saturates (exponentially quickly).

Type 1 temporal subspaces, on the other hand, do not lead to unconditional stability of the form (21). To formalize this, we define the CFL number

$$\text{CFL}_h := \max \Delta \mathcal{T}_E \sup_{\chi_h \in V_h \setminus \{0\}} \frac{\|\chi_h\|_V}{\|\chi_h\|_{V'}}. \quad (22)$$

Proposition 2 *Let $\{0\} \neq V_h \subset V$ be a finite-dimensional subspace. Let $E \subset H^1(J)$ and $F \subset L^2(J)$ be of Type 1. Then there exists a constant $\gamma_0 > 0$ independent of V_h , E and F , such that the discrete inf-sup condition (19) holds for the discrete trial and test spaces (17) with*

$$\gamma_h \geq \gamma_0 \kappa_h \min\{1, \text{CFL}_h^{-1}\}. \quad (23)$$

In general, the dependence on the CFL number cannot be improved.

4 Minimal residual Petrov-Galerkin discretization

In this section we consider an abstract continuous bilinear form $B : X \times Y \rightarrow \mathbb{R}$, where X and Y are Hilbert spaces with norms $\|\cdot\|_X$ and $\|\cdot\|_Y$. Let

$$\|B\| := \sup_{w \in X \setminus \{0\}} \sup_{v \in Y \setminus \{0\}} \frac{B(w, v)}{\|w\|_X \|v\|_Y}$$

denote the norm of B . Further, let b be a linear continuous functional of Y . For the remainder of the section, two finite-dimensional subspaces $X_h \subset X$ and $Y_h \subset Y$ are fixed. We aim at relaxing the discrete variational formulation (18) to admit the case $\dim X_h \lesssim \dim Y_h$. To guarantee well-posedness, the discrete inf-sup condition of B on $X_h \times Y_h$ will be essential (cf. Proposition 1):

$$\gamma_h := \inf_{w_h \in X_h \setminus \{0\}} \sup_{v_h \in Y_h \setminus \{0\}} \frac{B(w_h, v_h)}{\|w_h\|_X \|v_h\|_Y} > 0. \quad (24)$$

For practical reasons we introduce norms $\|\cdot\|_X$ and $\|\cdot\|_Y$ defined on X_h and Y_h that are induced by (positive definite) linear continuous operators $M : X_h \rightarrow X'$ and $N : Y_h \rightarrow Y'$ as follows:

$$\|w_h\|_X^2 := (Mw_h)(w_h), \quad w_h \in X_h, \quad (25)$$

$$\|v_h\|_Y^2 := (Nv_h)(v_h), \quad v_h \in Y_h. \quad (26)$$

The operators M and N are moreover assumed to be symmetric, i.e., $(Mu_h)(w_h) = (Mw_h)(u_h)$ for all $w_h, u_h \in X_h$, and similarly for N . Let $0 < d_M \leq D_M$ and $0 < d_N \leq D_N$ be constants such that

$$d_M \|\cdot\|_X \leq \|\cdot\|_X \leq D_M \|\cdot\|_X \quad \text{and} \quad d_N \|\cdot\|_Y \leq \|\cdot\|_Y \leq D_N \|\cdot\|_Y \quad (27)$$

on X_h and Y_h , respectively. We emphasize that the operators M and N , the induced norms, and hence the constants in (27) may depend on h , but in this section, $X_h \times Y_h$ is being considered fixed to lighten the notation.

Instead of the usual discrete variational formulation we now introduce the discrete (functional) residual minimization problem

$$u_h := \arg \min_{w_h \in X_h} R_h(w_h), \quad (28)$$

where

$$R_h(w_h) := \sup_{v_h \in Y_h \setminus \{0\}} \frac{|B(w_h, v_h) - b(v_h)|}{\|v_h\|_Y}, \quad w_h \in X_h, \quad (29)$$

is the (functional) residual. The following can be shown [4].

Theorem 1 *Let the discrete inf-sup condition (24) hold. Then there exists a unique $u_h \in X_h$ for which*

$$R_h(u_h) \leq R_h(w_h) \quad \forall w_h \in X_h \quad (30)$$

holds. Moreover, u_h satisfies the quasi-optimality estimate

$$\|u - u_h\|_X \leq C_h \inf_{w_h \in X_h} \|u - w_h\|_X \quad \text{with} \quad C_h = \frac{\|B\|}{\gamma_h} \frac{D_N}{d_N} \quad (31)$$

for any solution $u \in X$ of “ $B(u, \cdot) = b$ on Y ”.

Proof (Sketch) Invoking the open mapping theorem, one can show that the mapping $b \mapsto u_h$ is well-defined, linear and continuous, and its norm is dominated by $\frac{1}{\gamma_h} \frac{D_N}{d_N}$. Thus the composition $u \mapsto Bu \mapsto u_h$ is a continuous projection with norm not exceeding C_h given in (31). An application of [19, Lemma 5] finishes the argument.

Having introduced the somewhat nonstandard variational definition (28) of the discrete solution, let us describe its computable algebraic equivalent. To that end let $\Phi \subset X_h$ and $\Psi \subset Y_h$ be bases for the respective discrete spaces. Having fixed the pair $X_h \times Y_h$, the possible dependence on h is again omitted. The algebraic representants of the system bilinear form B , of the load functional b and of the norm-inducing operators N and M , are defined with respect to the chosen basis in the usual way,

$$\mathbf{B} := B(\Phi, \Psi), \quad \mathbf{b} := b(\Psi), \quad \mathbf{N} := (N\Psi)(\Psi), \quad \mathbf{M} := (M\Phi)(\Phi), \quad (32)$$

or in componentwise notation $\mathbf{B}_{\psi\phi} = B(\phi, \psi)$, $\mathbf{b}_\psi = b(\psi)$, $\mathbf{N}_{\psi'\psi} = (N\psi)(\psi')$, $\mathbf{M}_{\phi'\phi} = (M\phi)(\phi')$ for $\phi, \phi' \in \Phi$ and $\psi, \psi' \in \Psi$. The matrix \mathbf{B} is injective if and only if the discrete inf-sup condition (24) holds; further, \mathbf{N} and \mathbf{M} are symmetric positive definite matrices due to the analogous properties (27) of the operators N and M . Using these definitions, the discrete functional residual minimization (28) can be seen to be equivalent to the discrete algebraic residual minimization

$$\mathbf{u} := \arg \min_{\mathbf{w} \in \mathbb{R}^\Phi} \|\mathbf{B}\mathbf{w} - \mathbf{b}\|_{\mathbf{N}^{-1}}. \quad (33)$$

A vector \mathbf{u} is a stationary point of (33) if and only if it satisfies the first order conditions, namely the Gauß normal equations

$$\mathbf{B}^\top \mathbf{N}^{-1} \mathbf{B} \mathbf{u} = \mathbf{B}^\top \mathbf{N}^{-1} \mathbf{b}. \quad (34)$$

Since the residual error is measured in the $\|\cdot\|_{\mathbf{N}^{-1}}$ norm, we will refer to (34) as the *generalized* Gauß normal equations.

If the discrete inf-sup condition (24) holds, the matrix $\mathbf{B}^\top \mathbf{N}^{-1} \mathbf{B}$ is symmetric positive definite; then, the Gauß normal equations (34), and therefore also the discrete minimization problem (33), have a unique solution. Finally, the matrices \mathbf{M} and $\mathbf{B}^\top \mathbf{N}^{-1} \mathbf{B}$ are spectrally equivalent with [3, Section 4.1]

$$\gamma_h d_M d_N \|\mathbf{w}\|_{\mathbf{M}} \leq \|\mathbf{w}\|_{\mathbf{B}^\top \mathbf{N}^{-1} \mathbf{B}} \leq \|B\| D_M D_N \|\mathbf{w}\|_{\mathbf{M}} \quad \forall \mathbf{w} \in \mathbb{R}^\Phi. \quad (35)$$

Therefore, \mathbf{M} is a natural preconditioner for the Gauß normal equations (34). By (35), the quality of this preconditioner is controlled by the discrete inf-sup constant γ_h in (24) and the norm equivalence constants in (27), and does not depend on the choice of the basis.

5 Parabolic space-time preconditioners

In Section 4 we admitted general norm-inducing operators M and N on the fixed pair of finite-dimensional discrete trial and test spaces $X_h \times Y_h$. For the space-time variational formulation (5) several practical choices are available [3, Chapter 6]. Here, to simplify the exposition, we will only use the canonical choice of the Riesz mappings defined (on all of X and Y) by

$$(Mw)(w) := \|w\|_X^2 = \|w\|_{L^2(J;V)}^2 + \|\partial_t w\|_{L^2(J;V')}^2, \quad w \in X, \quad (36)$$

and

$$(Nv)(v) := \|v\|_Y^2 = \|v_1\|_{L^2(J;V)}^2 + \|v_2\|_H^2, \quad v = (v_1, v_2) \in Y. \quad (37)$$

These definitions extend to the off-diagonal by the imposed symmetry of M and N , but this will not play a role in the following. In these formulas, the $V = H_{\Gamma_0}^1(D)$ and the $V' = [H_{\Gamma_0}^1(D)]'$ norms are taken to be the “energy norms”:

$$\|\sigma\|_V^2 := (a \operatorname{grad} \sigma, \operatorname{grad} \sigma)_D, \quad \sigma \in V, \quad (38)$$

$$\|\varphi\|_{V'}^2 := (\varphi, A^{-1} \iota \varphi)_D, \quad \varphi \in H = L^2(D), \quad (39)$$

where A is the operator $A : V \rightarrow V'$, $\sigma \mapsto (a \operatorname{grad} \sigma, \operatorname{grad} \cdot)_D$, and $\iota \varphi \in V'$ is the functional on V defined by $\iota \varphi := (\varphi, \cdot)_D$, $\varphi \in H$. For a functional $\beta \in V'$, the definition extends to $\|\beta\|_{V'}^2 := \beta(A^{-1} \beta)$, but this will be irrelevant in the implementation below.

6 Kronecker product structure of the discretized operators

6.1 The system bilinear form

Recall from Section 2 the definition of the system bilinear form

$$B(u, v) := \int_J (\partial_t u, v_1)_D dt + \int_J (a \operatorname{grad} u, \operatorname{grad} v_1)_D dt + (u(0, \cdot), v_2)_D$$

for the space-time variational formulation of the model parabolic evolution equation, where $u \in X = L^2(J; V) \cap H^1(J; V')$ and $v = (v_1, v_2) \in Y = L^2(J; V) \times H$. As described in Section 3, we consider two types of discrete trial and test spaces. In either case these have the form

$$X_h = E \otimes V_h \subset X \quad \text{and} \quad Y_h = (F \otimes V_h) \times V_h \subset Y, \quad (40)$$

where $E \subset H^1(J)$ is the space of continuous piecewise affine functions on a temporal mesh, $F \subset L^2(J)$ is the space of piecewise constant functions on the same mesh (Type 1) or on its n -fold uniform refinement (Type 2), and $V_h \subset V$ is a finite-dimensional subspace.

For the remainder of the section we fix the spatial discretization $V_h \subset V$ with a basis $\Sigma \subset V_h$, and the temporal mesh $\mathcal{T}_E = \{0 = t_0 < t_1 < \dots < t_K = T\}$. Let $\mathcal{T}_F = \{0 = t'_0 < t'_1 < \dots < t'_{K'} = T\}$ be either \mathcal{T}_E or any n -fold uniform refinement of \mathcal{T}_E . Let E be as above, and let F denote the space of piecewise constant functions

with respect to the temporal mesh $\mathcal{T}_F \supseteq \mathcal{T}_E$, which possibly refines \mathcal{T}_E . As basis for E we take the usual hat functions $\Theta := \{\theta_k : k = 0, \dots, K\} \subset E$ defined by $\theta_k(t_{\tilde{k}}) = \delta_{k\tilde{k}}$. In particular, the only function that does not vanish at $t = 0$ is θ_0 . As basis for F we take the indicator functions $\Xi := \{\xi_k := \chi_{(t'_{k-1}, t'_k)} : k = 1, \dots, K'\}$ on the elements (t'_{k-1}, t'_k) of the temporal mesh $\mathcal{T}_F \supseteq \mathcal{T}_E$. These univariate bases are first combined to the collections $\Phi \subset X$, $\Psi_1 \subset Y_1$ and $\Psi_2 \subset Y_2$ as

$$\Phi := \{\theta \otimes \sigma : \theta \in \Theta, \sigma \in \Sigma\}, \quad \Psi_1 := \{\xi \otimes \sigma : \xi \in \Xi, \sigma \in \Sigma\}. \quad (41)$$

These now yield bases

$$\Phi \subset X_h \quad \text{and} \quad \Psi := (\Psi_1 \times \{0\}) \cup (\{0\} \times \Sigma) \subset Y_h \quad (42)$$

for the discrete trial and test spaces X_h and Y_h . Discretizing the bilinear form B using these tensor product bases Φ and Ψ as described in abstract terms in Section 4 leads to

$$\mathbf{B} = \begin{pmatrix} \mathbf{C}_t^{FE} \otimes \mathbf{M}_x + \mathbf{M}_t^{FE} \otimes \mathbf{A}_x \\ \mathbf{e}_t^E \otimes \mathbf{M}_x \end{pmatrix} \quad (43)$$

where **a**) the “temporal FEM” matrices $\mathbf{C}_t^{FE}, \mathbf{M}_t^{FE} \in \mathbb{R}^{\Xi \times \Theta}$ and the row vector $\mathbf{e}_t^E \in \mathbb{R}^\Theta$, have the components

$$[\mathbf{C}_t^{FE}]_{\xi\theta} = \int_J \theta'(t) \xi(t) dt, \quad [\mathbf{M}_t^{FE}]_{\xi\theta} = \int_J \theta(t) \xi(t) dt, \quad [\mathbf{e}_t^E]_\theta = \delta_{\theta_0\theta}, \quad (44)$$

with the prime denoting the derivative with respect to t , and **b**) the usual “spatial FEM” mass and stiffness matrices $\mathbf{M}_x, \mathbf{A}_x \in \mathbb{R}^{\Sigma \times \Sigma}$ are given by

$$[\mathbf{M}_x]_{\tilde{\sigma}\sigma} = \int_D \tilde{\sigma}(x) \sigma(x) dx, \quad [\mathbf{A}_x]_{\tilde{\sigma}\sigma} = \int_D a(x) \text{grad } \tilde{\sigma}(x) \cdot \text{grad } \sigma(x) dx. \quad (45)$$

Let us comment on the assembly of the temporal FEM matrices. First, if $\mathcal{T}_E = \mathcal{T}_F$ (Type 1) then $[\mathbf{C}_t^{FE}]_{\xi\theta} \in \{1, -1, 0\}$ and $[\mathbf{M}_t^{FE}]_{\xi\theta} \in \{\frac{1}{2}|I|, 0\}$ depending on whether ξ and θ are both nonzero on the temporal element I of \mathcal{T}_E having length $|I|$, and on the sign of θ' there. Therefore, assume now that \mathcal{T}_F is obtained from \mathcal{T}_E by a succession of uniform refinements (Type 2). Let \mathcal{T}_E^* denote the first uniform refinement of \mathcal{T}_E , and let E^* be the space of continuous piecewise affine function on \mathcal{T}_E^* with the hat function basis Θ^* . Let $\mathbf{C}_t^{FE^*}$ and $\mathbf{M}_t^{FE^*}$ denote the matrices as above, but with \mathcal{T}_E^* in place of \mathcal{T}_E . Consider now the embedding operator $S^E : E \rightarrow E^*$. Define the prolongation matrix $[\mathbf{S}_t^E]_{\theta^*\theta}$ by $S^E \theta = \sum_{\theta^* \in \Theta^*} \mathbf{S}_{\theta^*\theta}^E \theta^*$, $\theta \in \Theta$. Then

$$\mathbf{C}_t^{FE} = \mathbf{C}_t^{FE^*} \mathbf{S}_t^E \quad \text{and} \quad \mathbf{M}_t^{FE} = \mathbf{M}_t^{FE^*} \mathbf{S}_t^E. \quad (46)$$

Moreover, denoting by $t_\theta \in \mathcal{T}$ the node for which $\theta(t_\theta) = 1$, and similarly for $t_{\theta^*} \in \mathcal{T}^*$, we have “ $\mathcal{T}_E^* = \mathbf{S}_t^E \mathcal{T}_E$ ”, i.e.,

$$t_{\theta^*} = \sum_{\theta \in \Theta} [\mathbf{S}_t^E]_{\theta^*\theta} t_\theta \quad \forall \theta^* \in \Theta^*. \quad (47)$$

6.2 The norm-inducing operators

With the norms on V and V' taken to be (38)–(39), the discretized operators M and N defined in Section 5 assume the form

$$\mathbf{M} = \mathbf{M}_t^E \otimes \mathbf{A}_x + \mathbf{A}_t^E \otimes (\mathbf{M}_x \mathbf{A}_x^{-1} \mathbf{M}_x) \quad (48)$$

with the “temporal mass and stiffness” matrices

$$[\mathbf{M}_t^E]_{\tilde{\theta}\theta} = \int_J \tilde{\theta}(t) \theta(t) dt, \quad [\mathbf{A}_t^E]_{\tilde{\theta}\theta} = \int_J \tilde{\theta}'(t) \theta'(t) dt, \quad (49)$$

and

$$\mathbf{N} = \begin{pmatrix} \mathbf{M}_t^F \otimes \mathbf{A}_x & \\ & \mathbf{M}_x \end{pmatrix} \quad (50)$$

with the “temporal mass matrix”

$$[\mathbf{M}_t^F]_{\tilde{\xi}\xi} = \int_J \tilde{\xi}(t) \xi(t) dt. \quad (51)$$

6.3 Data structures

The Kronecker product structure of these matrices suggests regarding a vector $\mathbf{w} \in \mathbb{R}^{\Sigma \times \Theta}$ as a rectangular array with $\#\Sigma$ rows and $\#\Theta$ columns. Let $\text{Vec}(\mathbf{w})$ denote the “vectorization” of such an array, i.e., its columns are collected one after another into one column vector $\text{Vec}(\mathbf{w})$ of length $\#(\Sigma \times \Theta)$. Now, if $\mathbf{T} \in \mathbb{R}^{\Theta \times \Theta}$ and $\mathbf{X} \in \mathbb{R}^{\Sigma \times \Sigma}$ are matrices then

$$(\mathbf{T} \otimes \mathbf{X}) \text{Vec}(\mathbf{w}) = \text{Vec}(\mathbf{X} \mathbf{w} \mathbf{T}^\top). \quad (52)$$

In the implementation we will exclusively use the representation as rectangular arrays. Moreover, load vectors derived from load functionals $d \in Y'$ will be stored as pairs $\mathbf{d} = (\mathbf{d}^1, \mathbf{d}^2)$ with $\mathbf{d}^1 \in \mathbb{R}^{\Sigma \times \Xi}$ (rectangular array with $\#\Sigma$ rows and $\#\Xi$ columns) and $\mathbf{d}^2 \in \mathbb{R}^\Sigma$ (column vector of length $\#\Sigma$) in the form of a Matlab structure $\{\mathbf{d}1, \mathbf{d}2\}$. To these, a formula analogous to (52) applies. In particular, we never store the operators \mathbf{B} , \mathbf{M} and \mathbf{N} (or its inverses) as matrices.

7 Inverses of the space-time parabolic preconditioners

Consider the matrix representations \mathbf{M} and \mathbf{N} of the space-time parabolic preconditioners given in (48) and (50) in Section 6. In order to solve the generalized Gauß normal equations (34) with \mathbf{M} as preconditioner, we need to (approximately) compute the inverses \mathbf{M}^{-1} and \mathbf{N}^{-1} .

7.1 The test side

For \mathbf{N} we simply use the representation

$$\mathbf{N}^{-1} = \begin{pmatrix} (\mathbf{M}_t^F)^{-1} \otimes \mathbf{A}_x^{-1} \\ \mathbf{M}_x^{-1} \end{pmatrix}, \quad (53)$$

which again has Kronecker product structure. For problems with a large number of spatial degrees of freedom, the inverse \mathbf{A}_x^{-1} may be replaced by any approximate inverse, e.g. several cycles of a multigrid method.

7.2 The trial side

The representation of \mathbf{M}^{-1} requires more discussion, as it is not (obviously) of Kronecker product structure. We will obtain a simplified expression for \mathbf{M}^{-1} by diagonalizing \mathbf{M}_t^E and \mathbf{A}_t^E . For this discussion, let us drop the superscript $(\cdot)^E$. Consider therefore the generalized eigenvalue problem of finding $\mathbf{v} \in \mathbb{R}^\Theta$ and $\lambda \in \mathbb{R}$ such that $\mathbf{A}_t \mathbf{v} = \lambda \mathbf{M}_t \mathbf{v}$ (in place of \mathbf{M}_t , one could use the mass-lumped version of \mathbf{M}_t , or simply the diagonal matrix that coincides with \mathbf{M}_t on the diagonal). Let \mathbf{I}_t denote the identity matrix of the same size as \mathbf{M}_t and \mathbf{A}_t . Since \mathbf{A}_t is symmetric positive semi-definite and \mathbf{M}_t is symmetric positive definite, all eigenvalues are nonnegative and the eigenvectors may be chosen to form an \mathbf{M}_t -orthogonal basis: there exists a square matrix \mathbf{V}_t collecting the eigenvectors in its columns, and a diagonal matrix \mathbf{D}_t containing the eigenvalues on the diagonal, such that

$$\mathbf{V}_t^\top \mathbf{M}_t \mathbf{V}_t = \mathbf{I}_t \quad \text{and} \quad \mathbf{A}_t \mathbf{V}_t = \mathbf{M}_t \mathbf{V}_t \mathbf{D}_t. \quad (54)$$

Let us set $\mathbf{T}_t := \mathbf{M}_t \mathbf{V}_t$. The first identity implies $\mathbf{V}_t^{-1} = \mathbf{V}_t^\top \mathbf{M}_t = \mathbf{T}_t^\top$, which may be used to verify

$$\mathbf{M}_t = \mathbf{T}_t \mathbf{T}_t^\top \quad \text{and} \quad \mathbf{A}_t = \mathbf{T}_t \mathbf{D}_t \mathbf{T}_t^\top. \quad (55)$$

Inserting these into the expression (48) for \mathbf{M} we find

$$\mathbf{M} = (\mathbf{T}_t \otimes \mathbf{I}_x) (\mathbf{I}_t \otimes \mathbf{A}_x + \mathbf{D}_t \otimes (\mathbf{M}_x \mathbf{A}_x^{-1} \mathbf{M}_x)) (\mathbf{T}_t^\top \otimes \mathbf{I}_x) \quad (56)$$

and therefore

$$\mathbf{M}^{-1} = (\mathbf{V}_t \otimes \mathbf{I}_x) (\mathbf{I}_t \otimes \mathbf{A}_x + \mathbf{D}_t \otimes (\mathbf{M}_x \mathbf{A}_x^{-1} \mathbf{M}_x))^{-1} (\mathbf{V}_t^\top \otimes \mathbf{I}_x). \quad (57)$$

Let γ_θ be the square root of the entry of \mathbf{D}_t on the diagonal in position θ , i.e., $\mathbf{D}_t = \text{diag}((\gamma_\theta^2)_{\theta \in \Theta})$. Now, recall from Section 6 the convention that $\mathbf{w} \in \mathbb{R}^{\Sigma \times \Theta}$ is stored as a rectangular array with $\#\Theta$ columns. Applying \mathbf{M}^{-1} to such a vector \mathbf{w} will be done as follows

1. Compute $\mathbf{w}^{(1)} := \mathbf{w} \mathbf{V}_t$.
2. For each column $\mathbf{w}_\theta^{(1)}$ of $\mathbf{w}^{(1)}$ compute the corresponding column $\mathbf{w}_\theta^{(2)}$ of $\mathbf{w}^{(2)}$ by

$$\mathbf{w}_\theta^{(2)} := (\mathbf{A}_x + \gamma_\theta^2 (\mathbf{M}_x \mathbf{A}_x^{-1} \mathbf{M}_x))^{-1} \mathbf{w}_\theta^{(1)}. \quad (58)$$

3. Compute $\mathbf{w}^{(3)} := \mathbf{w}^{(2)} \mathbf{V}_t^\top$.

Then $\mathbf{M}^{-1}\mathbf{w} = \mathbf{w}^{(3)}$. The computation (58) can be done in parallel over the columns; we will further make use of the identity

$$(\mathbf{A}_x + \gamma_\theta^2(\mathbf{M}_x \mathbf{A}_x^{-1} \mathbf{M}_x))^{-1} = \overline{\mathbf{H}_\mp^{-1} \mathbf{A}_x \mathbf{H}_\mp^{-1}} = \mathbf{H}_\pm^{-1} \overline{\mathbf{A}_x \mathbf{H}_\pm^{-1}} \quad (59)$$

where $\mathbf{H}_\pm := \mathbf{A}_x \pm i\gamma_\theta \mathbf{M}_x$ is precisely the FEM discretization of the Helmholtz operator $(A \pm i\gamma_\theta \text{Id})$ with imaginary frequency $\pm i\gamma_\theta$. In fact, since the input is real, it would suffice to use $\text{Re } \mathbf{H}_\pm^{-1}$ instead of (59). Interestingly, such Helmholtz problems appeared in the context of (parabolic) evolution equations in [18, 8], but in the present case only in the representation of the preconditioner \mathbf{M}^{-1} . These may therefore be inverted approximately.

8 Assembly of the space-time load vector

As in the previous section, let F be the space of piecewise constant functions on a temporal mesh \mathcal{T}_F ; let Ξ be the basis on F consisting of indicator functions on the elements of \mathcal{T}_F ; finally, $\Sigma \subset V_h$ is a basis for a finite-dimensional subspaces $V_h \subset V = H_{I_0}^1(D)$. The load functional $b(v_1, v_2)$ defined in Section 2 can be rewritten as

$$b(v_1, 0) + b(0, v_2) = \int_J \{(f, v_1)_D + (g, v_1)_{\Gamma_N}\} dt + (h, v_2)_D, \quad (60)$$

for $(v_1, v_2) \in Y = L^2(J; V) \times H$, where $h \in H = L^2(D)$, $f \in L^2(J; H)$ and $g \in L^2(J; H^{1/2}(\Gamma_N))$ are given functions. Accordingly, the load vector \mathbf{b} that we obtain as outlined in Section 4 consists of two parts, say, $\mathbf{b}^1 \in \mathbb{R}^{\Sigma \times \Xi}$ and $\mathbf{b}^2 \in \mathbb{R}^\Sigma$. The second part is given in componentwise notation by

$$[\mathbf{b}^2]_\sigma = (h, \sigma)_D + (0, \sigma)_{\Gamma_N}, \quad \sigma \in \Sigma, \quad (61)$$

which is the usual spatial FEM load vector for the function $h \in L^2(D)$ with zero Neumann data. In the remainder of this section we describe how \mathbf{b}^1 may be obtained from the usual spatial FEM load vectors. Observe first that, fixing an indicator function $\xi \in \Xi$ supported on the closed interval I , we may employ a quadrature rule on I to define

$$[\mathbf{b}^1]_{\sigma\xi} := \sum_{r \in \mathbb{N}} w_r^I \left\{ (f(t_r^I, \cdot), \sigma)_D + (g(t_r^I, \cdot), \sigma)_{\Gamma_N} \right\} \quad (62)$$

$$\approx \int_I \{(f, v_1)_D + (g, v_1)_{\Gamma_N}\} dt, \quad \sigma \in \Sigma, \quad (63)$$

where $t_r^I \in I$ are the quadrature nodes and $w_r^I \in \mathbb{R}$ are the quadrature weights (equal to zero for $r \in \mathbb{N}$ large enough). Now, each term in the curly brackets $\{\dots\}$ is a load vector for the function $f(t_r^I, \cdot)$ with Neumann data $g(t_r^I, \cdot)$, which can be assembled using standard spatial FEM routines. This presupposes sufficient regularity of the functions $t \mapsto f(t, \cdot)$ and $t \mapsto g(t, \cdot)$ on the interval I . The computation of individual columns of \mathbf{b} (i.e., for each given $\xi \in \Xi$) can be performed in parallel.

If $\mathcal{T}_F = \mathcal{T}_E$ then using the trapezoidal rule on each temporal element I leads to a system $\mathbf{B}\mathbf{u} = \mathbf{b}$ that admits a unique solution, which on the nodes of \mathcal{T}_E coincides with the solution obtained by the Crank-Nicolson time-stepping method [14, 1].

9 Generalized LSQR algorithm

In Section 4 the minimal residual Petrov-Galerkin discretization was shown to lead to a system of generalized Gauß normal equations. Efficient iterative Krylov subspace methods for such problems are known [11]. One option is the LSQR algorithm due to Paige and Saunders [16] applied directly to the preconditioned equation $\tilde{\mathbf{B}}^T \tilde{\mathbf{B}} \tilde{\mathbf{u}} = \tilde{\mathbf{B}}^T \tilde{\mathbf{b}}$ with $\tilde{\mathbf{B}} = \mathbf{N}^{-1/2} \mathbf{B} \mathbf{M}^{-1/2}$, $\tilde{\mathbf{u}} = \mathbf{M}^{-1/2} \mathbf{u}$, $\tilde{\mathbf{b}} = \mathbf{N}^{-1/2} \mathbf{b}$, where $\mathbf{M}^{-1/2}$ and $\mathbf{N}^{-1/2}$ denote the inverses of the square roots of the (s.p.d.) matrices \mathbf{M} and \mathbf{N} . We now reformulate this version in such a way that only the inverses \mathbf{M}^{-1} and \mathbf{N}^{-1} need to be applied, but not the square roots, cf. [9]. To compute an approximate solution $\mathbf{u}_{i^*} \approx \mathbf{u}$ to the generalized Gauß normal equations (34) using \mathbf{M} as a preconditioner, the algorithm proceeds as follows:

1. Initialize
 - (a) $\mathbf{d}_0 := \mathbf{0}$
 - (b) $(\hat{\mathbf{v}}_0, \mathbf{v}_0, \beta_0) := \text{NORMALIZE}(\mathbf{b}, \mathbf{N})$
 - (c) $(\hat{\mathbf{w}}_0, \mathbf{w}_0, \alpha_0) := \text{NORMALIZE}(\mathbf{B}^T \hat{\mathbf{v}}_0, \mathbf{M})$
 - (d) $\rho_0 := \|(\alpha_0, \beta_0)\|_2$
 - (e) $\mathbf{u}_0 := \mathbf{0}$
 - (f) $\delta_1 = \alpha_0, \gamma_1 = \beta_0$
2. For $i = 1, 2, \dots, i^*$ do the following steps (until convergence)
 - (a) $\mathbf{d}_i := \hat{\mathbf{w}}_{i-1} - (\alpha_{i-1} \beta_{i-1} / \rho_{i-1}^2) \mathbf{d}_{i-1}$
 - (b) $(\hat{\mathbf{v}}_i, \mathbf{v}_i, \beta_i) := \text{NORMALIZE}(\mathbf{B} \hat{\mathbf{w}}_{i-1} - \alpha_{i-1} \mathbf{v}_{i-1}, \mathbf{N})$
 - (c) $(\hat{\mathbf{w}}_i, \mathbf{w}_i, \alpha_i) := \text{NORMALIZE}(\mathbf{B}^T \hat{\mathbf{v}}_i - \beta_i \mathbf{w}_{i-1}, \mathbf{M})$
 - (d) $\rho_i := \|(\delta_i, \beta_i)\|_2$
 - (e) $\mathbf{u}_i := \mathbf{u}_{i-1} + (\delta_i \gamma_i / \rho_i^2) \mathbf{d}_i$
 - (f) $\delta_{i+1} := -\delta_i \alpha_i / \rho_i, \gamma_{i+1} := \gamma_i \beta_i / \rho_i$

Here, $\text{NORMALIZE} : (\mathbf{s}, \mathbf{S}) \mapsto (\hat{\mathbf{z}}, \mathbf{z}, z)$, with \mathbf{S} s.p.d., is the procedure:

1. Solve $\mathbf{S} \hat{\mathbf{s}} = \mathbf{s}$ for $\hat{\mathbf{s}}$
2. Set $z := \sqrt{\mathbf{s}^T \hat{\mathbf{s}}}$ and $(\hat{\mathbf{z}}, \mathbf{z}) := (z^{-1} \hat{\mathbf{s}}, z^{-1} \mathbf{s})$

As long as the order of the statements is unchanged, the subscripts $(\cdot)_i$, etc., can be ignored in the implementation. In our implementation we will limit the number of iterations and allow the iteration to exit when the normal equations residual $\|\tilde{\mathbf{B}}^T \tilde{\mathbf{B}} \tilde{\mathbf{u}} - \tilde{\mathbf{B}}^T \tilde{\mathbf{b}}\|_2 = \|\mathbf{B}^T \mathbf{N}^{-1} \mathbf{B} \mathbf{u}_i - \mathbf{B}^T \mathbf{N}^{-1} \mathbf{b}\|_{\mathbf{M}^{-1}}$ falls below a tolerance threshold. This residual is available for each $i = 0, 1, \dots$ as $|\delta_{i+1}| \gamma_{i+1}$ following step (f). See [10] for further discussion of stopping criteria for the LSQR algorithm.

10 Overview of the Matlab code

In the code given below the naming convention parallels that of Sections 6–8. Thus, the mesh \mathcal{T}_F is called `TF`, the matrix \mathbf{M}_t^{FE} is called `MtFE`, the vector \mathbf{u} is called `u`, and so on. The subroutines that are related to the temporal FEM are prefixed with “`femT_`”, those related to spatial FEM with “`femX_`”. The only subroutine that mixes temporal and spatial FEM is `femTX_assemLoad` for the assembly of the space-time load vector `b`.

10.1 Main file

We provide the commented code for the main file. The code is embedded in a Matlab function `spacetime`.

```
1 function spacetime
```

Initialize the spatial FEM (load the mesh, etc., here into global variables) and compute the spatial FEM mass and stiffness matrices. The flag `true` indicates that this is the first-time initialization.

```
2 femX_init(true)
3 [Mx, Ax] = femX_MA();
```

Define a temporal mesh with K elements on the interval $J = (0, T)$ with $T = 4$:

```
4 T = 4; K = 100;
5 TE = T * sort([0, rand(1, K-1), 1]);
```

Determine the number of uniform refinements to go from \mathcal{T}_E to \mathcal{T}_F , see Section 6. Setting `use_mrpq = true` amounts to one refinement.

```
6 use_mrpq = true;
```

Compute the refined mesh \mathcal{T}_F and the temporal FEM matrices as described in Section 6:

```
7 [MtFE, CtFE, TF] = femT_assemFE(TE, use_mrpq);
8 [MtE, AtE] = femT_assemE(TE);
9 MtF = femT_assemF(TF);
```

Define the function that computes $\mathbf{w} \mapsto \mathbf{B}\mathbf{w}$ using the matrix representation given in Section 6:

```
10 function Bw = B(w)
11     Bw = { Mx * w * CtFE' + Ax * w * MtFE', Mx * w(:,1) };
12 end
```

Define the function that computes $\mathbf{v} \mapsto \mathbf{B}^T \mathbf{v}$ using the matrix representation given in Section 6.

```
13 function Btv = Bt(v)
14     Btv = Mx' * v{1} * CtFE + Ax' * v{1} * MtFE;
15     Btv(:,1) = Btv(:,1) + Mx' * v{2};
16 end
```

Define the function that computes $\mathbf{d} \mapsto \mathbf{N}^{-1} \mathbf{d}$ with \mathbf{N}^{-1} as in Section 7:

```
17 function iNd = iN(d)
18     iNd = { Ax \ (d{1} / MtF), Mx \ d{2} };
19 end
```

Define the function that computes $\mathbf{w} \mapsto \mathbf{M}^{-1} \mathbf{w}$ using the algorithm and formula (59) given in Section 7. Two comments are in order. First, the symmetric positive semi-definite matrix \mathbf{A}_t^E is singular, possibly leading to a small negative approximately computed eigenvalue. Before taking the square root we therefore round negative eigenvalues to zero. Second, the result of an application of (59) to a real vector is again a real vector, which is enforced by taking the `real` part. This truncates the round-off error accumulated in the imaginary part and reestablishes the data type of reals. The loop can be performed in parallel.

```

20 [VtE, DtE] = eig(full(AtE), full(MtE));
21 gamma = sqrt(max(0, diag(DtE)));
22 function iMw = iM(w)
23     iMw = w * VtE;
24     parfor j = 1:length(TE)
25         H = Ax + 1i * gamma(j) * Mx;
26         iMw(:,j) = real(H \ conj(Ax * (H \ iMw(:,j))));
27     end
28     iMw = iMw * VtE';
29 end

```

Compute the space-time load vector \mathbf{b} using a quadrature rule according to Section 8. The Matlab functions `f`, `g` and `h` are assumed to be available, e.g. as Matlab files in the same directory. Further, `QR_Trapz` is the trapezoidal quadrature rule as described in Section 10.2.

```

30 b = femTX_assemLoad(TF, @f, @g, @h, QR_Trapz());

```

Set the tolerance and the maximal number of iterations for the generalized LSQR algorithm from Section 9 and run it on the Gauß normal equations (34) with \mathbf{M} as preconditioner. The solver may provide additional diagnostic output parameters that are ignored here.

```

31 tol = 1e-8; maxit = 100;
32 u = glsqr(@B, @Bt, b, tol, maxit, @iM, @iN);

```

Finally, plot several temporal snapshots (equispaced in time) of the numerical solution. These are obtained by linear interpolation of the values at time nodes and stored in the array `u`.

```

33 U = interp1(TE, u', linspace(min(TE), max(TE), 5))';
34 for k = 1:size(U,2)
35     subplot(1, size(U,2), k); femX_show(U(:,k));
36 end

```

Here the Matlab function `spacetime` ends.

```

37 end

```

10.2 Assembly of the space-time load vector

The assembly of the space-time load vector \mathbf{b} is performed in the Matlab function `femTX_assemLoad`. It receives the mesh \mathcal{T}_F as a (row) vector `TF`, as well as function handles `f`, `g` and `h`. The function handles `f` and `g`, when called with one argument, say t_r , are expected to return function handles to functions that depend on the spatial variable only and describe $f(t_r, \cdot)$ and $g(t_r, \cdot)$. The function handle `h` describes the initial condition $h(\cdot)$. Finally, `QuadRule` is a function handle that receives a 2-component vector describing the endpoints of an interval I and returns a quadrature rule on I in the form of a vector of nodes $(t_r^I)_{r=1,\dots,R}$ and a vector of corresponding weights $(w_r^I)_{r=1,\dots,R}$, as well as the number of nodes R .

```

1 function b = femTX_assemLoad(TF, f, g, h, QuadRule)

```

Assemble the part \mathbf{b}^2 of the load vector from the initial condition $h(\cdot)$.

```

2 b2 = femX_b(h, @(varargin)0);

```

Assemble \mathbf{b}^1 by iterating over the temporal elements defined by \mathcal{T}_F as described in Section 8. The outer loop can be computed in parallel.

```

3  b1 = zeros(length(b2), length(TF)-1);
4  parfor k = 1:size(b1,2)
5      [tI, wI, R] = QuadRule(TF([k k+1]));
6      for r = 1:R
7          b1(:,k) = b1(:,k) + wI(r) * femX_b(f(tI(r)), g(tI(r)));
8      end
9  end

```

The assembled vectors \mathbf{b}^1 and \mathbf{b}^2 are combined into a Matlab structure.

```
10  b = {b1, b2};
```

Here the Matlab function `femTX_assemLoad` ends.

```
11 end
```

10.3 Assembly of the temporal FEM matrices 1

Let us comment in some detail on the computation of the temporal FEM matrices \mathbf{C}_t^{FE} and \mathbf{M}_t^{FE} by means of the Matlab function `femT_assemFE`. It receives a temporal mesh $\mathcal{T}_0 \supseteq \mathcal{T}_E$ and the number `nref` of uniform refinements to be performed on \mathcal{T}_0 to obtain \mathcal{T}_F . If `nref` is interpreted as one or zero if it has the logical value `true` or `false`, respectively. The output is \mathbf{C}_t^{FE} and \mathbf{M}_t^{FE} , and the `nref`-fold refinement of \mathcal{T}_0 stored again in the variable `T0`.

```
1 function [MtFE, CtFE, T0] = femT_assemFE(T0, nref)
```

If no refinement is to be performed, the temporal FEM matrices \mathbf{C}_t^{FE} and \mathbf{M}_t^{FE} can be computed directly. We remark that temporal meshes are stored as row vectors.

```

2  K = length(T0);
3  if (nref == 0)
4      MtFE = spdiags(diff(T0)' * [1/2 1/2], 0:1, K-1, K);
5      CtFE = diff(speye(K));
6      return
7  end

```

Otherwise, the prolongation matrix \mathbf{S}_t^E is computed first, see Section 6.1. As can be seen from (47), it coincides with the matrix representation of an interpolation operator (a more efficient but lengthier implementation is possible here).

```
8  StE = sparse(interp1(1:K, eye(K), 1:(1/2):K));
```

Perform a uniform refinement of the current mesh \mathcal{T}_0 and pass the new mesh recursively to `femT_assemFE`. Hence, the number of refinements still to be performed decreases by one. We obtain the matrices \mathbf{M}_t^{FE*} and \mathbf{C}_t^{FE*} with respect to the meshes \mathcal{T}_F and \mathcal{T}_0^* , and the `nref`-th refinement of \mathcal{T}_0 (which is the desired \mathcal{T}_F).

```
9  [MtFEs, CtFEs, T0] = femT_assemFE((StE * T0)'), nref-1);
```

Apply the prolongation matrix according to (46) to obtain \mathbf{M}_t^{FE} and \mathbf{C}_t^{FE} .

```
10  MtFE = MtFEs * StE; CtFE = CtFEs * StE;
```

Here the Matlab function `femT_assemFE` ends.

```
11 end
```


10.4 Assembly of the temporal FEM matrices 2

The computation of the temporal stiffness and mass matrices \mathbf{A}_t^E , \mathbf{M}_t^E and \mathbf{M}_t^F is a routine task, and the Matlab code is therefore provided for completeness without further comment.

```

1 function [MtE, AtE] = femT_assemE(TE)
2   K = length(TE); h = diff(TE); g = 1./h;
3   MtE = spdiags([h 0; 0 h]' * [1 2 0; 0 2 1]/6, -1:1, K, K);
4   AtE = spdiags([g 0; 0 g]' * [-1 1 0; 0 1 -1], -1:1, K, K);
5 end

1 function MtF = femT_assemF(TF)
2   K = length(TF)-1;
3   MtF = sparse(1:K, 1:K, abs(diff(TF)));
4 end

```

10.5 Example

Let us set $a = 1$ for the heat conduction coefficient and $f(t, x) = \sin(t)$ for the source term in (1), $g = 0$ for the Neumann data (3) and $h = 0$ for the initial condition (4). Using the well-documented open-box 2d spatial FEM discretization from [2, Sections 2–8] the code described in this section (and some graphical postprocessing) produces a figure similar to Fig. 1.

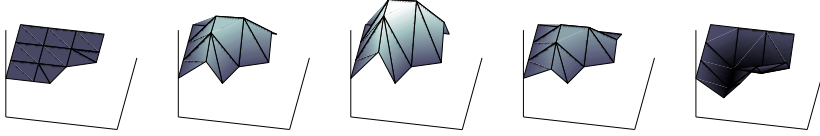


Fig. 1 Snapshots of the solution at $t = 0, 1, 2, 3, 5$, produced by the code given in Section 10. Dark corresponds to low, bright to high values.

Continuing with these parameters, we set $T = 20$. In Fig. 2 we document the accuracy of the discrete solution and the number of iterations of the generalized least squares solver for equidistant temporal meshes \mathcal{T}_E of different size and different numbers of refinements between \mathcal{T}_E and \mathcal{T}_F . We observe that for Type 1 temporal subspaces (no refinement; this method is equivalent to the Crank-Nicolson time-stepping), the number of iterations first increases with $\#\mathcal{T}_E$ but then decreases again. For Type 2 temporal subspaces (one or more refinements), the number of iterations varies with $\#\mathcal{T}_E$ much less and is consistently smaller. Taking the number of iterations as an indicator of the conditioning of the generalized normal equations with \mathbf{M} as preconditioner and taking (35) into account, these observations are in agreement with Propositions 1 and 2. Replacing the trapezoidal rule by a higher-order quadrature does not significantly change that picture. Note that this is not the number of iterations need to reach the accuracy of the same order of

magnitude as admitted by the temporal discretization, but the number of iterations needed to achieve a fixed small residual in the preconditioned Gauß normal equations as described in Section 9.

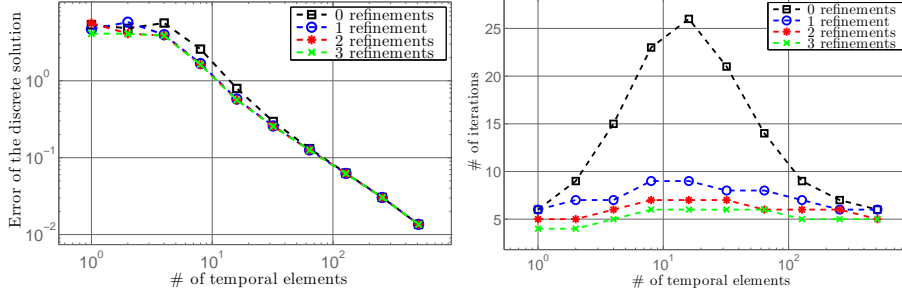


Fig. 2 Accuracy of the discrete solution in the space-time norm $\|\cdot\|_X$ (left) and the number of generalized LSQR iterations (right) as a function of the number of temporal elements, see Section 10.5.

11 Conclusions

A concise Matlab implementation of a space-time simultaneous discretization and solution algorithm for parabolic evolution equations that is stable, parallelizable, and modular has been presented. It admits nonuniform temporal meshes and time-dependent inputs. Very efficient natural preconditioners for the iterative resolution of the resulting single linear system of equations are available. Extensions to higher order in time and to space-time compressive algorithms are possible [4, 3, 5].

Let us point out what we believe is currently the major obstruction to massively parallel computations based on the algorithm presented here. Allowing arbitrary temporal meshes, the simultaneous diagonalization of the temporal FEM matrices discussed in Section 7 leads to a nonlocal-in-time transformation \mathbf{V}_t . If the solution vector is split over multiple computational nodes along the temporal dimension, which is natural, the application of this transformation may result in heavy communication. If, however, the temporal mesh is uniform, or derives from a successive dyadic partition of a coarse mesh, it may be possible to use more efficient fast Fourier or wavelet based transforms for this particular step instead.

Acknowledgements Research in part supported by the Swiss NSF Grant No. 127034 and by the ERC AdG No. 247277 awarded to Ch. Schwab.

References

1. Akrivis, G., Makridakis, C., Nochetto, R.: Galerkin and Runge–Kutta methods: unified formulation, a posteriori error estimates and nodal superconvergence. *Numer. Math.* **118**, 429–456 (2011)
2. Albery, J., Carstensen, C., Funken, S.A.: Remarks around 50 lines of Matlab: short finite element implementation. *Numer. Algorithms* **20**(2-3), 117–137 (1999)
3. Andreev, R.: Stability of space-time Petrov–Galerkin discretizations for parabolic evolution equations. Ph.D. thesis, ETH Zürich (2012). ETH Diss. No. 20842
4. Andreev, R.: Stability of sparse space-time finite element discretizations of linear parabolic evolution equations. *IMA J. Numer. Anal.* (2012). Available online
5. Andreev, R., Tobler, C.: Multilevel preconditioning and low rank tensor iteration for space-time simultaneous discretizations of parabolic PDEs. Tech. rep., ETH Zürich (2012). Submitted
6. Babuška, I., Janik, T.: The h-p Version of the Finite Element Method for Parabolic Equations. I. The p Version in Time. *Numer. Methods Partial Differential Equations* **5**, 363–399 (1989)
7. Babuška, I., Janik, T.: The h-p Version of the Finite Element Method for Parabolic Equations. II. The h-p Version in Time. *Numer. Methods Partial Differential Equations* **6**, 343–369 (1990)
8. Banjai, L., Peterseim, D.: Parallel multistep methods for linear evolution problems. *IMA J. Numer. Anal.* **32**(3), 1217–1240 (2012)
9. Benbow, S.J.: Solving generalized least-squares problems with LSQR. *SIAM J. Matrix Anal. Appl.* **21**(1), 166–177 (electronic) (1999)
10. Chang, X.W., Paige, C.C., Tittley-Peloquin, D.: Stopping criteria for the iterative solution of linear least squares problems. *SIAM J. Matrix Anal. Appl.* **31**(2), 831–852 (2009)
11. Choi, S.C.: Iterative methods for singular linear equations and least-squares problems. Ph.D. thesis, Stanford University (2006)
12. Evans, L.C.: Partial Differential Equations, *Graduate Studies in Mathematics*, vol. 19. American Mathematical Society (1998)
13. Fattorini, H.O.: Infinite dimensional linear control systems, *North-Holland Mathematics Studies*, vol. 201. Elsevier Science B.V., Amsterdam (2005)
14. Hulme, B.L.: One-step piecewise polynomial Galerkin methods for initial value problems. *Math. Comp.* **26**, 415–426 (1972)
15. Lions, J.L., Magenes, E.: Non-homogeneous boundary value problems and applications. Vol. I. Springer-Verlag, New York (1972)
16. Paige, C.C., Saunders, M.A.: LSQR: an algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Software* **8**(1), 43–71 (1982)
17. Schwab, C., Stevenson, R.: Space-time adaptive wavelet methods for parabolic evolution problems. *Math. Comp.* **78**(267), 1293–1318 (2009)
18. Sheen, D., Sloan, I.H., Thomée, V.: A parallel method for time discretization of parabolic equations based on Laplace transformation and quadrature. *IMA J. Numer. Anal.* **23**(2), 269–299 (2003)
19. Xu, J., Zikatanov, L.: Some observations on Babuška and Brezzi theories. *Numer. Math.* **94**(1), 195–202 (2003)